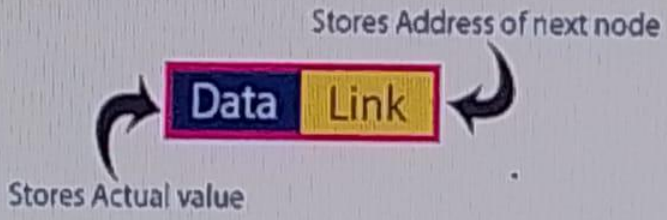
1. Explain the operations of a Singly Linked List with suitable examples.
2. List different hash functions used in hashing, and explain any two of them with examples.
3. Define a Queue. Write algorithms for enqueue() and dequeue() operations, and illustrate them with examples.
4. What are the three possible cases to consider while deleting a node from a Binary Search Tree (BST)?
5. What is a Skip List? Explain the insertion operation in a skip list with an example.

Single Linked List:

Simply a list is a sequence of data, and the linked list is a sequence of data linked with each other.
The formal definition of a single linked list is as follows...

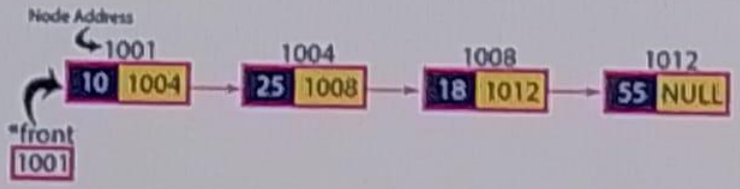
Single linked list is a sequence of elements in which every element has link to its next element in the sequence.

In any single linked list, the individual element is called as "Node". Every "Node" contains two fields, data field, and the next field. The data field is used to store actual value of the node and next field is used to store the address of next node in the sequence. The graphical representation of a node in a single linked list is as follows...



In a single linked list, the address of the first node is always stored in a reference node known as "front" (Some times it is also known as "head").
Always next part (reference part) of the last node must be NULL

Example



1a

2a. HASHING:

- Hashing is the process of mapping large amount of data item to a smaller table with the help of a hashing function.
- The Value Generated by Hash function is called "Hash value" which is used as an index to store values in the Hash Table.

dictionary entries in the hash table using hash function.

HASH FUNCTION:

- Hash function is a function which maps key values to array indices. (OR) · Hash Function is a function which, when applied to the key, produces an integer which can be used as an address in a hash table. ·
- We will use h(k) for representing the hashing function Hash Values: The values returned by a hash function are called hash values or hash codes or hash sums or simply hashes

  For example: Consider that we want place some employee records in the hash table The record of employee is placed with the help of key: employee ID. The employee ID is a 7 digit number for placing the record in the hash table. To place the record 7 digit number is converted into 3 digits by taking only last three digits of the key. If the key is 496700 it can be stored at 0th position. The second key 8421002, the record of those key is placed at 2nd position in the array.

  Hence the hash function will be- H(key) = key%1000 Where key%1000 is a hash function and key obtained by hash function is called hash key.

  Bucket and Home bucket: The hash function H(key) is used to map several dictionary entries in the hash table. Each position of the hash table is called bucket.

Types of hash function

There are various types of hash function which are used to place the data in a hash table,

1. Division method

In this the hash function is dependent upon the remainder of a division

H(key)=key%tablesize

For example table size is 5 ,keys are 10,18,24,26 then

H(10)=10%5=0-->this value used as index to store value in the hash table

H(15)=18%5=3

H(24)=24%5=4

H(26)=26%5=1

0	10
1	26
2	
3	18
4	24

2. Mid square method

In this method firstly key is squared and then mid part of the result is taken as the index. For example: consider that if we want to place a record of 3101 and the size of table is 1000. So 3101*3101=9616201 i.e. **h (3101) = 162 (middle 3 digit)**

3. Digit folding method

In this method the key is divided into separate parts and by using some simple operations these parts are combined to produce a hash key. For example: consider a record of 12465512 then it will be divided into parts i.e. 124, 655, 12. After dividing the parts combine these parts by adding it.

```
H(key)=124+655+12
      =791
```

Multiplication method: The given record is multiplied by some constant value. The formula for computing the hash keyis-
H(key) = floor (tablesize(key*A)%tablesize))where p is integer constant, and A is constant realnumber.

Donald Knuth suggested to use constant A = 0.61803398987
If key 23 and tablesize=10 then

H(key) = floor (10*(23*0.61803398987)%10
= floor (142.14)%10
= 142

At 142 locations in the hash table the record 23 will be placed.

QUEUES:

- Queue is a linear data structure where the first element is inserted from one end called **REAR** and deleted from the other end called as **FRONT**.
- **Front** points to the **beginning** of the queue and **Rear** points to the **end** of the queue.
- Queue follows the **FIFO (First - In - First Out)** structure.
- According to its FIFO structure, element inserted first will also be removed first.
- In a queue, one end is always used to insert data (enqueue) and the other is used to delete data (dequeue), because queue is open at both its ends.
- The enqueue() and dequeue() are two important functions used in a queue.

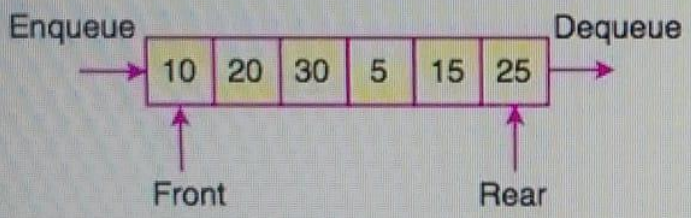


Fig. Queue

3a

Basic Operations of Queue:

- **Enqueue:** This operation is used to Insert an element at the rear end of the queue.
- **Dequeue:** This operation is used to remove and return an element from the front end of the queue.

enQueue(value) - Inserting an element into the Queue

We can use the following steps to insert a new node into the queue...

- **Step 1** - Create a **newNode** with given value and set '**newNode → next**' to **NULL**.
- **Step 2** - Check whether queue is **Empty** (**rear == NULL**)
- **Step 3** - If it is **Empty** then, set **front** = **newNode** and **rear** = **newNode**.
- **Step 4** - If it is **Not Empty** then, set **rear → next** = **newNode** and **rear** = **newNode**.

deQueue() - Deleting an Element from Queue

We can use the following steps to delete a node from the queue...

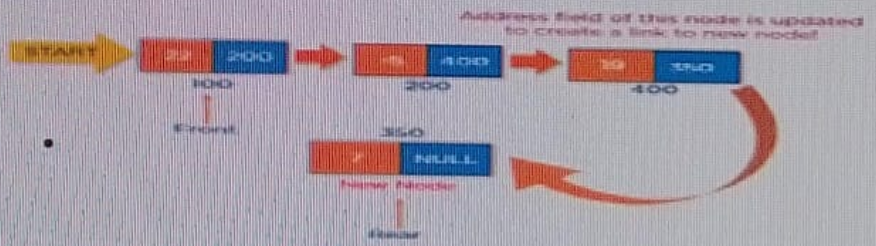
- **Step 1** - Check whether **queue** is **Empty** (**front == NULL**).
- **Step 2** - If it is **Empty**, then display **"Queue is Empty!!! Deletion is not possible!!!"** and terminate from the function
- **Step 3** - If it is **Not Empty** then, define a Node pointer '**temp**' and set it to '**front**'.
- **Step 4** - Then set '**front** = **front → next**' and delete '**temp**' (**free(temp)**).

display() - Displaying the elements of Queue

We can use the following steps to display the elements (nodes) of a queue...

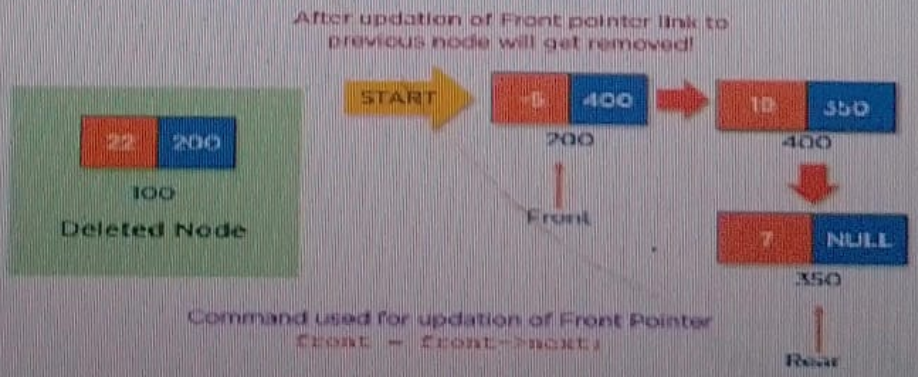
- **Step 1 -** Check whether queue is **Empty** (**front** == **NULL**).
- **Step 2 -** If it is **Empty** then, display **'Queue is Empty!!!'** and terminate the function.
- **Step 3 -** If it is **Not Empty** then, define a Node pointer **'temp'** and initialize with **front**.
- **Step 4 -** Display '**temp** → **data** --->' and move it to the next node. Repeat the same until '**temp**' reaches to '**rear**' (**temp** → **next** != **NULL**).
- **Step 5 -** Finally! Display '**temp** → **data** ---> **NULL**'.

Enqueue:



- The value of the rear pointer will become 350 now, whereas the front pointer remains the same.

- **Dequeue**: After deleting an element from the linked queue, the value of the front pointer will change from 100 to 200. The linked queue will look like below:



Binary Search Tree Deletion:

1. Delete function is used to delete the specified node from a binary search tree.

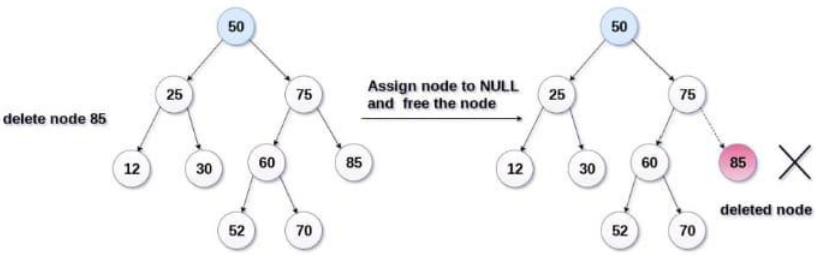
4a)

2. First we have to check with deleted node with root node if is less than root node then we have to check left side of the tree if node to be deleted is greater than parent node we have to check right side of the tree.
3. There are three situations of deleting a node from binary search tree.

1)Node to be deleted is leaf node:

It is the simplest case, in this case, replace the leaf node with the NULL and simple free the allocated space.

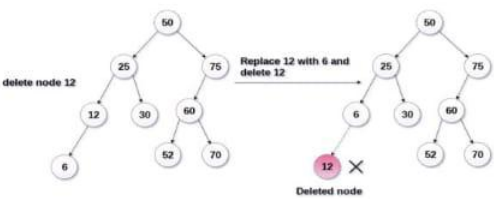
In the following image, we are deleting the node 85, since the node is a leaf node, therefore the node will be replaced with NULL and allocated space will be freed.



2)The node to be deleted has only one child:

In this case, replace the node with its child and delete the child node, which now contains the value which is to be deleted. Simply replace it with the NULL and free the allocated space.

In the following image, the node 12 is to be deleted. It has only one child. The node will be replaced with its child node and the replaced node 12 (which is now leaf node) will simply be deleted.



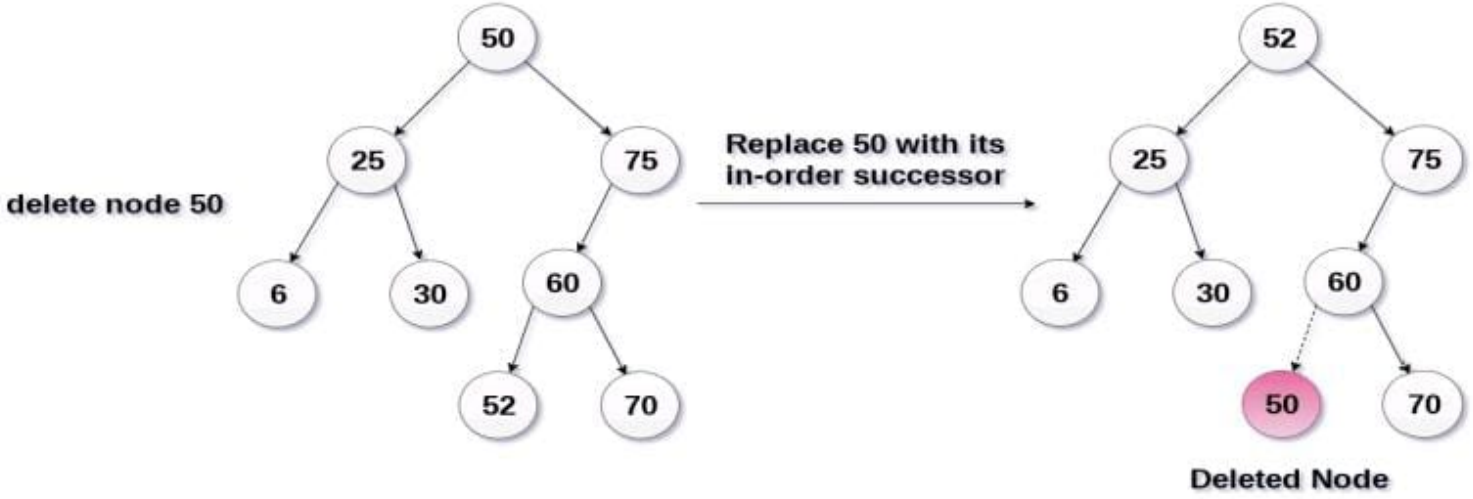
3)Node to be deleted has two children:

Node which is to be deleted has two children then we can replace that node with

3)Node to be deleted has two children:

Node which is to be deleted has two children then we can replace that node with

a. Inorder predecessor (largest element from the left subtree) OR
b. Inorder successor (smallest element from the right sub tree)



In the following image, the node 50 is to be deleted which is the root node of the tree.

The in-order traversal of the tree given below.

6, 25, 30, 50, 52, 60, 70, 75.

replace 50 with its in-order successor 52. Now, 50 will be moved to the leaf of the tree, which will simply be deleted.

Skip list in Data structure

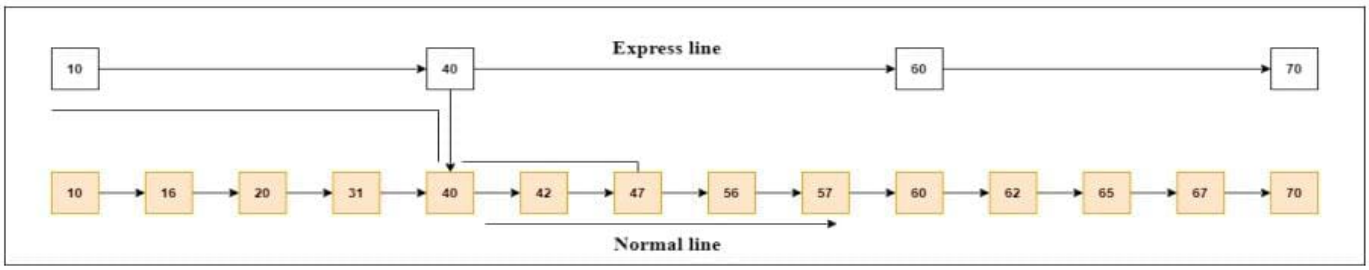
1)A skip list is a probabilistic data structure. The skip list is used to store a sorted list of elements or data with a linked list. It allows the process of the elements or data to view efficiently. In one single step, it skips several elements of the entire list, which is why it is known as a skip list.

2)The skip list is an extended version of the linked list. It allows the user to search, remove, and insert the element very quickly. It consists of a base list that includes a set of elements which maintains the link hierarchy of the subsequent elements.

Skip list structure **(5a)**

It is built in two layers: The lowest layer and Top layer.

The lowest layer of the skip list is a common sorted linked list, and the top layers of the skip list are like an "express line" where the elements are skipped.



Skip List Basic Operations

There are the following types of operations in the skip list.

1) Insertion operation: It is used to add a new node to a particular location in a specific situation.

2)Deletion operation: It is used to delete a node in a specific situation.

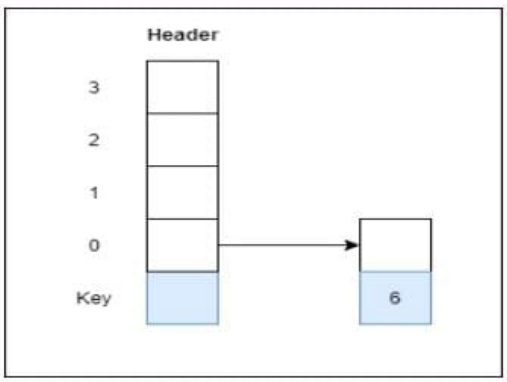
3)Search Operation: The search operation is used to search a particular node in a skip list.

Example 1: Create a skip list, we want to insert these following keys in the empty skip list.

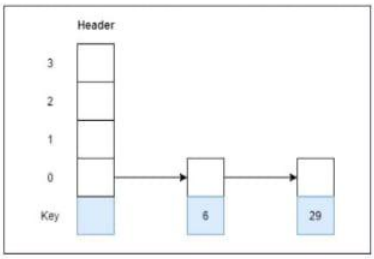
1. 6 with level 1.
2. 29 with level 1.
3. 22 with level 4.
4. 9 with level 3.
5. 17 with level 1.
6. 4 with level 2.

Ans:

Step 1: Insert 6 with level 1

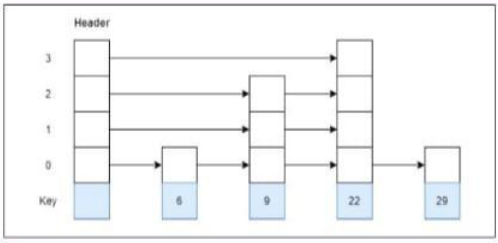


Step 2: Insert 29 with level 1

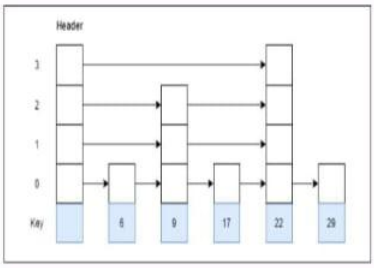


Step 3: Insert 22 with level 4

Step 4: Insert 9 with level 3



Step 5: Insert 17 with level 1



Step 6: Insert 4 with level 2